redmineorg-copy202205 - Vote #66519

Create and maintain groups from LDAP attributes

2022/05/09 14:29 - Admin Redmine

ステータス:	New	**開始日:**	2010/02/08
優先度:	通常	**期日:**	
担当者:		**進捗率:**	80%
カテゴリ:	LDAP_28	**予定工数:**	0.00時間
対象バージョン:		**作業時間:**	0.00時間
Redmineorg_URL:	https://www.redmine.org/issues/4755	**status_id:**	1
category_id:	28	**tracker_id:**	3
version_id:	0	**plus1:**	4
issue_org_id:	4755	**affected_version:**	
author_id:	11448	**closed_on:**	
assigned_to_id:	0	**affected_version_id:**	
comments:	13		

説明

We do have group information in LDAP which we would like to use. These are essentially

- ou (the organisational unit(s) the person is associated with)
- businessCategory (the type of user: employee, guest, …)

I decided to write the following patch to automatically create and update group information for every user whenever she logs in, in order to avoid having to extract all the information regularly from LDAP, keeping in the spirit of on-the-fy creation of user accounts.

h2. Challenges

During the implementation, I was facing the following problems:

- I wanted to have both attributes available for group creation, both independently and as a cross product (i.e., being able to add all employees or everyone from orgUnit1 to the project, but also have the option to add only the employees of orgUnit1 to some project)
- Our ou names are very long (some approaching 60 characters), but the lastname field of the User (which is where the group names are stored for the Group subclass/subtype of User) is limited to 30 characters. I did not want to change that part of the model, so I am shortening LDAP group names on import.

h2. Functionality implemented

There are 5 additional fields in @auth_source@:

attr_groups: The name of the attribute containing group information (in our case: ou)

attr_groups2: The name of an optional second attribute containing other group information (in our case: businessCategory)

** None, either, or both of the above can be empty, if they are not required

group_prefix: A character sequence you might want to prefix to the group names which are thus automatically generated (default: "_")

cross_product: When this is false, only the group names based on attr_groups and attr_groups2 are included. When this is set to true, *additionally* the

concatenation of group names derived from @attr_groups@ and @attr_groups2@ (separated by @group_separator@) are included

group_separator: The separator to use if cross-products should be created

Groups created like that will have their @auth_source@ set accordingly. Groups with their @auth_source@ set will have an appropriate note attached in the list and edit views, and their name is not editable. LDAP-maintained groups are automatically deleted when they are no longer necessary (triggered on login).

The group list view provides a "Refresh groups" button for each LDAP source with group update activated. It is not strictly necessary, as membership will be updated on the next login of that user anyway, but it might be helpful to see what effect your changes have or make membership updates visible immediately (as opposed to waiting for the next login of this user/these users).

The messages are available in English and German. Feel free to provide other languages as well :-).

h2. Code overview

Here is a short run-down of what the code provides:

Two migration files to add the required attributes

@auth_source_ldap.rb@ has the following new features

** a method get_attributes which fetches LDAP attributes for a user specified (does not test for authentication)
** a method build_names to construct the names of the groups that should be associated with
** Two classes, @FakeLdapCon@ and @FakeLdapEntry@ which are helpful when trying to experiment with LDAP attributes (do not require an LDAP server)

@user.rb@ has the following new features:

** Can store group_names temporarily (needed to get things working with a consistent implementation for normal login, auto-relogin, and on-the-fly creation of users and to save a few round-trips to LDAP)
** a method refresh_group_memberships to update the group membership information based on what was returned by LDAP

@group.rb@ was changed as follows:

** Contains a new class method, shorten_lastname, to (nicely, IMHO) shorten group names when the names in LDAP exceed the length limit
** Two new methods, size_and_updated_by_string and updated_by_string to simplify the form rendering routines

@auth_sources_controller.rb@ now includes a method for the "refresh groups" button described above

@auth_sources/list.rhtml@ and @auth_sources/_form.rhtml@ do provide UI for this button and the new group fields

@groups/index.html.erb@ and @groups/_form.html.erb@ do provide the UI changes for the automatic groups

@locales/en.yml@ and @locales/de.yml@ have the text elements for the new UI

@migrate/20100207220329_extend_ldap_groups.rb@ and @migrate/20100204211355_add_ldap_group_support.rb@ are the two new migrations (I did the implementation in two stages)

h2. The patch

The patch *includes* a better version of #4643 (Allow on-the-fly creation on member addition), as I was unhappy with the way I originally hacked @auth_source_ldap.rb@ (too much code duplication in my original version). If you do not want that feature, back out the change to @members_controller.rb@ after applying the patch. Also, the change to @auth_source.rb@ (new class methods import and get_data) is only strictly required when on-the-fly project member addition is needed.

The patch does *not* include #4732 (Make login case-insensitive also for PostgreSQL), although line numbers might reflect that change.

The patch is against 0.9.0rc.

h2. Trunk integration?

Jean-Philippe et al.: What is required to get this code integrated into trunk? I think that LDAP integration without something like this is severely limited and thus this should be included.

journals

Please use the ldap-auto-groups2.patch instead, as it fixes an issue with on-the-fly registration (LDAP query attribute list was computed wrongly, which would only show on the production system).

My internal Q&A needs to be improved :-)

+1

What you did is extremly close to our needs here at my institute. Thanks for your work! I would appreciate to have a function like that in trunk. Maybe you noticed some similar efforts, done in #1113.

I do like your way of customizing what LDAP attributes are used for creating redmine groups.

That's just guessing without having inspected your code, but maybe it's possible to be even more generic and give the Redmine admin an option to add as many attr_groups-like fields as he may need (with performance issues in mind ;-)). Maybe even setting up custom rules about how to connect information from different fields (AND/OR) with each other. Maybe it would be possible to choose a name for LDAP-sourced groups in Redmine (with some kind of hash or similar) if you won't like to use the names used in LDAP (however thats more a goody than really needed from my point of view).

On the one hand, being more generic would be more complicated on the coding-site and gives admins an opportunity to mess things up. On The other hand, being even more generic does increase chances, that group information can be fetched from a high amount of different LDAP/ActiveDirectory Installations without custom patches and therefore make your additions even more valuable ;-).

I have not been aware of #1113, thanks for pointing this out.

Indeed, there have been thoughts about having more than two fields to use for grouping. However, as it would complicate UI even further, I wanted to leave it at this stage until there is actual demand for >2 grouping criteria; instead have working code that I can deploy in our setup (we have deployed it to be a platform for anyone within the University with project/issue tracking needs). The lastname length limitation of 30 is already an issue when crossproducting two groups, I do think naming would become too complicated, also causing a huge explosion of cross-product entries (exponential in the number of dimensions).

We already have the flexibility, as the attribute names are configurable (per LDAP source). But I agree it would be great to have some LDAP group support in the trunk without the need for custom patches.

(A further idea I did put off until actual demand (and not just wishes) would be shown, were a translation table between LDAP and internal group names.)

+1

+1

Does this patch will be available for 1.x branch ?

This is the output with redmine-1.0.1 :

```
patching file models/auth_source_ldap.rb
Hunk #1 FAILED at 33.
Hunk #2 FAILED at 81.
Hunk #3 FAILED at 95.
3 out of 3 hunks FAILED -- saving rejects to file models/auth_source_ldap.rb.rej
patching file models/user.rb
Hunk #1 FAILED at 49.
Hunk #2 FAILED at 102.
Hunk #3 FAILED at 117.
Hunk #4 FAILED at 134.
Hunk #5 FAILED at 302.
Hunk #6 FAILED at 321.
6 out of 6 hunks FAILED -- saving rejects to file models/user.rb.rej
patching file models/auth_source.rb
Hunk #1 FAILED at 17.
Hunk #2 FAILED at 46.
2 out of 2 hunks FAILED -- saving rejects to file models/auth_source.rb.rej
patching file models/group.rb
Hunk #1 FAILED at 24.
Hunk #2 FAILED at 45.
2 out of 2 hunks FAILED -- saving rejects to file models/group.rb.rej
patching file controllers/members_controller.rb
Hunk #1 FAILED at 24.
1 out of 1 hunk FAILED -- saving rejects to file controllers/members_controller.rb.rej
patching file controllers/auth_sources_controller.rb
Hunk #1 FAILED at 72.
1 out of 1 hunk FAILED -- saving rejects to file controllers/auth_sources_controller.rb.rej
patching file views/auth_sources/list.rhtml
Hunk #1 FAILED at 20.
1 out of 1 hunk FAILED -- saving rejects to file views/auth_sources/list.rhtml.rej
patching file views/auth_sources/_form.rhtml
Hunk #1 FAILED at 39.
1 out of 1 hunk FAILED -- saving rejects to file views/auth_sources/_form.rhtml.rej
patching file views/groups/index.html.erb
Hunk #1 FAILED at 13.
1 out of 1 hunk FAILED -- saving rejects to file views/groups/index.html.erb.rej
patching file views/groups/_form.html.erb
Hunk #1 FAILED at 1.
1 out of 1 hunk FAILED -- saving rejects to file views/groups/_form.html.erb.rej
patching file locales/en.yml
Hunk #1 FAILED at 276.
Hunk #2 FAILED at 743.
Hunk #3 FAILED at 787.
Hunk #4 FAILED at 853.
4 out of 4 hunks FAILED -- saving rejects to file locales/en.yml.rej
patching file locales/de.yml
Hunk #1 FAILED at 279.
Hunk #2 FAILED at 693.
Hunk #3 FAILED at 732.
Hunk #4 FAILED at 781.
4 out of 4 hunks FAILED -- saving rejects to file locales/de.yml.rej
patching file schema.rb
Hunk #1 FAILED at 9.
Hunk #2 FAILED at 43.
Hunk #3 FAILED at 473.
3 out of 3 hunks FAILED -- saving rejects to file schema.rb.rej
patching file migrate/20100207220329_extend_ldap_groups.rb
patching file migrate/20100204211355_add_ldap_group_support.rb
```

Thanks

I do not have the time to update it right now. If there is a decision for "official" integration, I would be happy to update it, but right now, the Redmine Gods first need to be convinced to get this or #1113 into Redmine. All of those who need such a feature, please discuss here or in #1113 or in the forums to get things moving forward.

Hi,

This is a quick 'n' dirty patch working with 1.0.1 release.
If you need only one group attribute, then you must fill attr_groups2 with the same value as attr_groups.

Hope this will help.

+1

There is a new plugin for "ldap sync":
https://github.com/thorin/redmine_ldap_sync/blob/master/README.md

Hi all,

I have forked the plugin to take in account :

- Dynamic group members (*groupOfURLs* versus groupOfNames)
- Dynamic groups member cache (120 seconds by default)
- One single ldap connection for the rake task

- A dry-run mode

---

related_issues

relates,New,5742,Association of an LDAP group to a Redmine group
relates,Closed,5702,Please add ldap filters for authentication
relates,Closed,4643,Allow on-the-fly creation on member addition
relates,New,6202,On-the-fly group addition based on LDAP sources

履歴

#1 - 2022/05/10 17:23 - Admin Redmine

- カテゴリ を LDAP_28 にセット